PST-15418/36
40410sh

AN AUGMENTED COMPUTER USER LOGIN AUTHENTICATION USING CLASSIFYING REGIONS OF KEYSTROKE DENSITY NEURAL NETWORK

GOVERNMENT SPONSORSHIP

This work was supported by the United States Army under Contract No. DAAD19-01-1-0504. Accordingly, the U.S. government may have certain rights to this invention.

BACKGROUND OF THE INVENTION

Authentication of users is important to prevent unauthorized access to computer systems and networks. Many studies show keystroke dynamics can be used to uniquely identify users. A study, sponsored by National Bureau of Standards in 1980, and conducted by Stanford Research Institute for keystroke-based authentication reported 98% authentication when the users type in their ids and passwords alone [BP01]. In 1980, Rand Corporation concluded the practicality of typewriter keyboard dynamics. Barton and Barton, in 1984, suggested the use of keyboard characteristics in the determination of passwords [L91]. In 1985, observations about the keying patterns of telegraph operators showed each operator exhibited a distinct keying pattern (seen in [UW85]).

Leggett et al. [LW88], M. Brown et al. [BR93], Monrose and Rubin [MR97], Robinson et al. [RJ98], Leggett et al. [L91], Bleha et al. [BSH90] have developed methods based on keystroke dynamics for verification of users with successful results. Gaines et al. [GLPN80], and Young et al. [YH89] have patents for user identification based on keystroke dynamics. Products based on keystroke latencies, like ‘BioPassword,” promise 100% accurate authentication based on keystroke latency [BP01].

SUMMARY OF THE INVENTION

We present an authentication system using classifying regions of keystroke density based on a neural network architecture with two types of connections: (1) weight vector W and (2) dispersion vector V. In the learning phase, the weight vector W adapts to users’ keystroke exemplars, and dispersion vector V adapts to dispersion in the users’ keystrokes. Here W represents users’ keystroke pattern, and V represents the radius for the regions of density of

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
SEP 2005		**00-00-2005 to 00-00-2005**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
An Augmented Computer User Login Authentication Using Classifying Regions of Keystroke Density Neural Network	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Pennsylvania State University,Mechanical Engineering Department,University Park,PA,16802**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
PSU Invention Disclosure No. 2003-0780, Licensed with BioPassword Inc., Issaquah, WA 98027, September 2005

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**12**	

users' keystrokes. The system consists of three phases: (1) training, (2) validation, and (3) testing. The system learns W and V during training, and adjustment of parameters SF and PS (see Section 3) is done during validation. During testing, classification results in strengthening the vector W, thereby adapting to changing users' typing pattern. We achieved up to individual 0% IPR and 0% FAR. Our highest results are 1.36% IPR and 2.31% FAR. These results compare favorably to the results reported in the literature [BR93] [BSH90] [JG90] [LW88] [UW85].
The significant contributions of our approach outlined below.

- A novel technique for user classification. From the perspective of fundamental research, this work introduces new neural architecture. From the application perspective, it provides a new approach to build classification systems.
- Unique formation of input vector not considered in earlier studies.
- Variability-weights keep track of variability in the key-stroke patterns. Weights form an individual cluster-mean of each key-stroke component.
- The network adapts to the change in typing patterns because the NN learns during classification (no need to re-train the network as the typing characteristics change).
- Addition and deletion of users does not affect the NN connection for other users, resulting in no alteration of weights of existing users. This results in fast addition and deletion of users.
- Security can be enhanced or relaxed based on the Scale Factor (SF) and 'percentage-success' (PS) values.

DETAILED DESCRIPTION OF THE INVENTION

The rest of the paper is organized as follows: Section 2 explains the data collection and processing; Section 3 explains the system architecture; Section 4 gives the results; and Section 5 concludes the paper.

1. Data Collection and Processing

Users select their user-id any string with which they are familiar, such as name, email id, etc., because the speed and pattern of typing becomes consistent as one types a string repeated

number of times. We restricted the password to a minimum of six characters while no minimum length was enforced for user-id.

2.1 Construction of Input Vector

The timestamp of ‘key-press’ and ‘key-release’ of every letter is recorded as the reference signature. For example the reference signature for the substring ‘vir’ is recorded as “KP V 2937531 ; KR V 2937640 ; KP I 2937843 ; KR I 2937921 ; KP R 2938156 ; KR R 2938250 ;” . ‘KP’ stands for key-press, ‘V’ is the letter being pressed, and the 2937531 is the timestamp when it was pressed, followed by the delimiter ‘;’. Similarly ‘KR’ stands for key-release, ‘V’ is the letter being released, and the 2937640 is the timestamp when it was released, followed by the delimiter ‘;’.

From the timestamp of key-press time and key-release time of the characters typed the following three parameters are found: (1) ‘key-press time’, (2) ‘key-latency’ between key-press-timestamps, (3) and ‘key-latency’ between key-release-timestamps. Figure 1 shows the calculation of these three parameters. The ‘key-press’ time (KPT) is the amount of time (in milliseconds) a key is held pressed. Since KP and KR is being calculated, we have two different ‘key latency’ times, one is the time difference between two successive KP times, and the other is the time difference between two successive KR times. We name the former one as ‘key latency of key press time’ (KLKPT) and the later as ‘key latency of key release time’ (KLKRT).

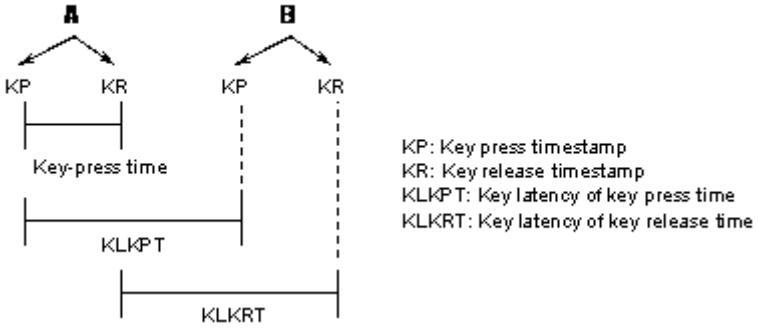

Figure 1. Key-press time and key-latency time calculations for “AB” key presses.

We do not consider all the KPT, KLKPT, and KLKRT of the reference signatures for feeding to the NN. Spaces between words are not considered [LW88] [UW85]. Only up to first

six characters of a word are considered [seen in UW85]. For the user-id seven key latency values are considered. For single worded user-id, first seven latencies between first eight characters are considered; for multiword user-id, five latencies from the first word and two from the second word are considered. If the latencies fall short of seven latencies due to user-id variation in length, then last latency is repeated until seven latencies are achieved. For example, in user-id string "sadeep moorthiyedath", the latencies are “sa, ad, de, ee, ep, mo, oo”. For user-id ‘vir phoha” the latencies are “vi, ir, ph, ho, oh, ha, ha”, the last latency is repeated twice to get seven latencies. Since there are KLKPT and KLKRT latencies, we have 14 latencies to consider, seven from each. For a string of length ‘n’ there exists ‘n-1’ latencies and ‘n’ key presses. Since user-id can have two words, we consider altogether nine key-press times. If the keys fall short of nine key-presses, the last key-press is repeated until nine. From the user-id we altogether consider 23 keystroke pattern values (nine key-presses and 14 key latencies).

Since password length is restricted to a minimum six characters, we consider all the six key press times and five key latency times making it altogether 16 key pattern values. For the common-string, ‘master of science in computer science’ nine latencies, first five latencies from the word ‘master’, one from ‘of’, and first three latencies from ‘science’ are considered. 12 key-press times, six from ‘master’, two from ‘of’, and first four from ‘science’ are considered. Altogether 30 key patterns are considered.

These different key patterns form a vector of 69 values and is obtained as follows. For the following ‘uid’ means user-id, ‘p’ means password, ‘cs’ means common-string. User id key pattern = {uid-KPT, uid-KPKPT, uid-KPKRT} which is of length 23 (9+7+7). Password-key-pattern = {p-KP, p-KPKPT, p-KPKRT} which is of length 16 (6+5+5). Common-string-key-pattern = {cs-KP, cs-KPKPT, cs-KPKRT} which is of length 30 (12+9+9). The complete key pattern consists of Key pattern vector = {uid key-pattern, password-key-pattern, common-string-key-pattern}. This Key-pattern vector is of length 69 (23=9+7+7, 16 = 6+5+5, 30) and forms the input to the NN, during training of reference signature and classification of user during login.

2.2 Selection of six best representative vectors

Out of the nine sample reference signatures, we choose six vectors as follows. We used six vectors, following the study ([JG90] see page 174, item 3), however the method to select these

vectors is our own. The NN with a one dummy output node is trained on all the 9 reference vectors. We sort the Euclidean distance between the input vectors X and weight vectors W_{dummy}, and choose the six closest, that is the six vectors which have the smallest Euclidean distance. It implies considering those reference signature that have less degree of variability among themselves. These 6 original vectors then form the inputs to the NN for training the particular user node.

2. Neural Network Architecture

The NN is a two layered network with '69' input nodes I and 'n' output nodes Y. Every input node is connected to every output node via weights w_{ij}, where 'i' is the input node and 'j' is the output node.

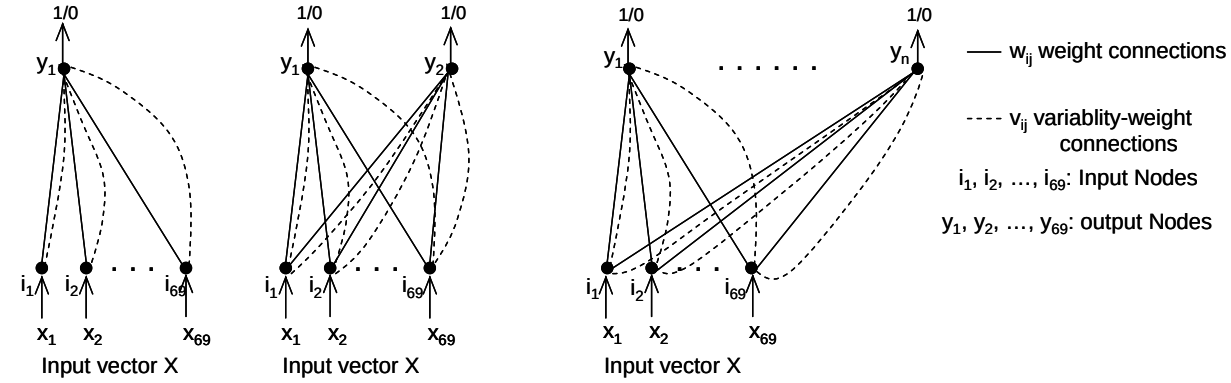

Figure 2. Neural Network Architecture of key-stroke classification system. Addition (or deletion) of users do not affect other users.

The weight vector $W_j = \{w_{1j}, w_{2j}, \ldots, w_{69,j}\}$ consists of all the weight connecting the input nodes to the output node 'j'. Apart from the weights, every input node is connected to every output node via variability-weights $V_j = \{v_{1j}, v_{2j}, \ldots, v_{69,j}\}$. Variability-weight v_{ij} connects i^{th} input node to j^{th} output node, and vector V_j consists of variability-weights connecting all input nodes to output node 'j'. The input vector X has 69 components. One output node is assigned per user, and additional output nodes can be added as and when new users register with the system. Figure 2(a) shows the weight connections of NN for one user, Figure 2(b), when another user is

added, and Figure 2(c) for n users.

Figure 3 shows the algorithm of the NN for training. During training, six input vectors ($T = \{t_{ij}\}$ $1 \le i \le 69$; $1 \le j \le 6$ and $T_k = \{t_{1k}, t_{2k}, \ldots, t_{69,k}\}$ [because there are six training vectors] of the user's keystroke patterns form the inputs. Weights are updated according to the rule

$$w_{ij} = w_{ij}(old) + \eta^{train}(T_k - W_j);$$

Where η^{train} is the learning factor used during training. η^{train} is decreased by α in each iteration during training. Training of 'variability-weights' proceed after the weights have been fully trained. The variability-weights are updated according to the rule

$$v_{ij} = \underset{k=1,6}{Max} \left| t_{ik} - w_{ij} \right|$$

where t_{ik} is the i^{th} input component of k^{th} vector. Initially the values of these 'variability-weights' are set to zero. Components 't_{ik}' are compared with its corresponding weight component 'w_{ij}' for user 'j', the current training user. If the absolute difference between them is greater than the present 'variability-weight' value, then this value is set to be the current 'variability-weight' value.

```
//Algorithm to train the NN.
Train_NN (array of Input vector T , user no 'j')
{
        W_j = T_1;                                  //Initialize the weight to first vector for faster convergence.
        Do                                          //Train the weights.
        {
                for each input vector T_i
                {
                        W_j = W_j(old) + η^train ( T_i - W_j); //update the weights.
                }
                 η^train = η^train * α ;            //decrease η^train by α
        }                                           //do until the weights so not show significant increase.
        While (difference in wt update is not greater than 'Error');
        //train the 'variability-weights'
        V_j = {0};                                  //initialize 'variability-weights' to zero.
        For each input vector T_i                   //repeat for each input vector.
        {
                for each component in the input     //k = 1 to 69
                {
                        diff = abs(t_ik - w_kj );   //find the absolute difference between the input and the current wij
                        if(diff is greater than v_kj)
                        {
                                v_kj = diff;        //set v_ij to current difference.
                        }
                }
        }
}//end of training.
```

Figure 3. NN algorithm for training a users' cluster given the array of reference patterns T_i.

Figure 4 shows the algorithm for classification phase. During classification a single vector G forms the input. The users' node Y_j is classified as true (1) by the following rule.

$$c_i = \begin{cases} 1 & if \left| g_i - w_{ij} \right| < SF * v_{ij} \\ 0 & otherwise \end{cases}$$

$$y_j = \begin{cases} 1 & if \left| G \right| / 69 >= PF \quad // user authenticated \\ 0 & otherwise \quad // not authenticated \end{cases}$$

Where |G| denotes the number of 1's in G.

Here C is a vector indicating the number of components of input that fall within the cluster boundary as defined by V_j and SF. We denote input vector as T (six samples) during training and G (one sample only) during classification. Each input component g_i is compared with the stored weight component w_{ij}, for j^{th} user, and its absolute difference value computed. This absolute value is compared with a scaled 'variability-weight'. If the difference is within the scaled 'variability-weight', then the key-pattern for this component is matched. The 'variability-weight' is scaled by 'scale-factor' times (SF). This value signifies the amount of variability to be allowed in user's typing pattern while classification. A higher value allows more variability in user's typing pattern, but also increases the chances of impersonation. A lower value indicates tightened typing behavior, making them to be same as entered in training.

The user is classified if his percentage of 'pattern matches' is greater than a preset value of 'percentage-success' (PS). That is if the 'percentage-success' is set to be 90% then the user's total individual pattern matches (g_i with scaled v_{ij}) must be 90% or greater. This factor allows the tightening of security of the system. A higher value results in lesser error but increases FAR. A lower value reduces the FAR while increasing error. If a user is successfully classified, then the weights of the NN are updated with $\eta^{classify}$, the value of η during classification.

```
bool ClassifyUser(Input vector G, user no 'j')
{
        count = 0;                              //initialize the counting variable.
        for each component gi in the input vector
        {
                diff = abs(gi - wij );          //find the absolute difference between the component and the weight component.
                if ( diff <= SF * vij  )        //check id this value falls within the range of the cluster.
                {
                        count++;                //keep count of this success.
                }
        }
        If( (count/69) >= 'percentage-success')         //check to see if percentage of this count is within the range specified.
        {
                authenticate = true;                    //authenticated the users.
                Wj = Wj(old) + η^classify( G - Wj);     //update the weights.
        }
        else
        {
                authenticate = false;           //user not authenticated.
        }
        return authenticate;                    //return true or false
}//end of classification.
```

Figure 4. NN algorithm for classifying (and adapting) a user given the key-pattern input vector G

3. RESULTS

A total of 43 users took part in providing reference signatures. Apart from providing reference signatures for training the NN, every user participated in providing login samples and impersonating others. A total of 873 login samples were collected, out of which 216 samples were authentic samples—users trying to login providing correct user-id and passwords, and 657 imposter samples—users trying to impersonate others. We use two measures during classification, Imposter Pass Rate (IPR) and False Authentication Rate (FAR). IPR is the ratio of successful impersonation samples over the total impersonation samples. FAR is the ratio of wrong rejection of authentic samples over total authentic samples.

Table 1 shows IPR and FAR as SF and PS are varied. IPR increases, as SF and PS are increased; FAR decreases faster when PS is reduced. The optimal result we get is an IPR of 1.36% (9 out of 657 samples) and an FAR of 2.31% (5 out of 216 samples). The best result is to have 0% IPR with least possible FAR. An ideal situation is to have minimum IPR with minimum FAR. In our experiment we achieve 0% IPR with 60% FAR. For an IPR of 0.3% (2 error out of

657) FAR is 18.5% (40 out of 216). We varied SF and PF from 0.1 to 3.0 in increments of 0.1 and validated the results during validation phase.

Table 2 shows the results with different combinations of user-id, password, and common string. Having a common-string and considering KPT, KLKPT, KLKRT decreases IPR with better FAR rates. Considering user-id, password, and common-strings with KP, KLKPT, and KLKRT yield better results for the same set of optimal parameters. Table 3 compares our results with the results of other methods.

Table 1. Varying results of IPR and FAR with PS and SF varied at $\eta^{classify} = 0.001$, $\eta^{train} = 0.4$, and $\alpha = 0.5$

SF	1.5	1.75	2	2.3	3	1.5	1.5	1.5	1.5
PS	0.9	0.9	0.9	0.9	0.9	0.85	0.8	0.75	0.7
IPR (%)	0	0	0.15	0.15	1.21	0.15	0.15	1.06	2.28
FAR (%)	89.9	73.61	49.53	38.88	12.5	56.01	36.11	7.87	2.31

Table 2. IPR and FAR for different types of experiments with parameters SF = 1.9, PS = 0.77, $\eta^{classify} = 0.001$, $\eta^{train} = 0.4$, and $\alpha = 0.6$

Experiment Type	Name, password, and Common-string	Name and password only	Name and password (KLKRT)	Name, Password, common-string (KLKRT)
IPR (%)	1.36	2.13	4.41	3.19
FAR (%)	2.31	18.98	29.62	14.35

Table 3. Comparison of results with other existing methods.

Method		Our Method	J. Leggett and G. Williams [LW88]	S Bleha, et al. [BSH90]	J. Leggett et al.	D. Umpress and G. Williams [UW85]	M. Brown and S. Rogers [BR93]	R. Joyce and G. Gupta [JG90]
Best result	IPR (%)	0.3 (2 out of 657)	-	-	-	-	0	0
	FAR (%)	18	-	-	-	-	12	40
Optimal Result	IPR (%)	1.36 (9 out of 657)	2.7	2.8	5	6	-	0.25
	FAR (%)	2.31	12.2	8.1	5.5	12	-	16.36

4. CONCLUSION

Significant features of our approach include (1) addition of dispersion vector to define keystroke regions, (2) adaptation to changing typing patterns, (3) ease in addition and deletion of users, and (4) unique derivation of input vectors. We are extending this work to define tighter boundaries for vector V that will result in improved results.

Abstract: We develop a system consisting of a neural architecture resulting in classifying regions corresponding to users' keystroke patterns. We extend the adaptation properties to classification phase resulting in learning of changes over time. Classification results on login attempts of 43 users (216 valid, 657 impersonation samples) show considerable improvements over existing methods.

5. REFERENCES

[BP01] “Bio Password.” Technical Report, BioPassword Keystroke Dynamics, October 2001. (Available from http://www.biopassword.com/home/technology/BP%204.5%20Technical%20Paper.pdf).

[BR93] M. Brown and S. J. Rogers. User Identification via Keystroke Characteristics of Typed Names using Neural Networks. International Journal of Man-Machine Studies, 39(6):999-1014, 1993.

[BSH90] S. Bleha, C. Slivinsky, and B. Hussein. Computer-Access Security Systems Using Keystroke Dynamics. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-12(12):1217-1222, December 1990.

[GLPN80] R. Gaines, W. Lisowski, S. Press, and Shapiro N. Authentication by Keystroke Timing: some preliminary results. Rand Report R-256-NSF. Rand Corporation, 1980.

[JG90] R. Joyce and G. Gupta, Identity Authentication Based on Keystroke Latencies, Communication of the ACM, Volume 33, Number 2, pp. 168-176, February 1990.

[LW88] J. Leggett and G. Williams. Verifying Identity via Keystroke Characteristics. International Journal of Man-Machine Studies, 28(1): 67-76, 1988.

[MR97] F. Monrose and A. D. Rubin: Authentication via Keystroke Dynamics. ACM Conference on Computer and Communications Security 1997: 48-56

[RJ98] J. Robinson, et al. “Computer user verification using login string keystroke dynamics.” IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, Vol. 28, No. 2, March 1998, pp. 236-241.

[UW85] D. Umphress and G. Williams. Identity Verification Through Keyboard Characteristics. International Journal of Man-Machine Studies, 23(3): 263-273, 1985.

[YH89] J.R. Young and R.W. Hammon. Method and apparatus for verifying an individual’s identity. Patent No. 4 805 222, U.S. Patent and Trademark Office, 1989.

EXEMPLARY CLAIMS

1. A method of authenticating the user of a secure system, comprising:

training a neural network to recognize the keystrokes of an authorized user; and

analyzing the keystrokes of a potential user to determine if they are authorized to use the system.

2. The method of claim 1, wherein neural network utilizes two types of connections: (1) weight vector W and (2) dispersion vector V; and

wherein in the training phase, the weight vector W adapts to users' keystroke exemplars, and dispersion vector V adapts to dispersion in the users' keystrokes.

3. The method of claim 1, wherein a user is asked to submit particular information to train the neural network.

4. The method of claim 3, wherein the particular information includes a predetermined string of keystrokes.